****

**PI Tag Value Checker Module Developer Manual**

**for the**

**I-Field Tags Synchronization System**

**at**

**EXPEC Computer Center (ECC)**

Saudi Aramco, Saudi Arabia

|  |
| --- |
|  |

**Document Type: Developer Manual**

**Document Name: ECC PI Tag Value Checker Developer Manual**

**Version: 1**

**Version Date: 11-May-2018**

**Crucial Solutions & Services (CSS)**

**Address**: P.O. Box 10381 | Jubail Industrial City 31961 | Kingdom of Saudi Arabia   
**Tel**: +966 (013) 349 3391 | **Fax**: +966 (013) 349 3392

**Email**: sales@css.net.sa | **Website**: [www.css.net.sa](http://www.css.net.sa)

Copyright © 2018 Crucial Solutions and Services (CSS)

The information contained herein is the sole property of Crucial Solutions & Services (CSS)**.** This document is reserved for viewing by EXPEC Computer Center (ECC), Saudi Aramco (Aramco) and consented recipients. No part of this document, may be reproduced, transmitted or modified (in any form or by any means) outside of the intended recipients without the express prior written consent of Crucial Solutions & Services (CSS).

Windows, SQL Server, SharePoint, Office, and Excel are registered trademarks of Microsoft Corporation.

ProcessBook, DataLink, WebParts, Coresight, AF, HA, and PI are registered trademarks of OSIsoft Inc.

All other trademarks are the property of their respective owners.

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Rev | Date | Updated by | Revision Comments |
| 1 | 11-May-2018 | Houssam Al Moughrabi | First Issue |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[SECTION 1: Summary 4](#_Toc513819401)

[SECTION 2: Architecture 5](#_Toc513819402)

[2.1 Diagram 5](#_Toc513819403)

[2.2 ECCPITagValueChecker\_Service - Windows Service 6](#_Toc513819404)

[2.3 ECC\_DataLayer - Data Layer 7](#_Toc513819405)

[2.3.1 Repositories 7](#_Toc513819406)

[2.3.2 Stores 7](#_Toc513819407)

[2.3.3 Helpers 8](#_Toc513819408)

[2.3.4 DataModels 8](#_Toc513819409)

[2.4 ECC\_ServicesLayer – Service Layer 9](#_Toc513819410)

[2.4.1 Services 9](#_Toc513819411)

[2.4.2 Helpers 10](#_Toc513819412)

[SECTION 3: Workflow 11](#_Toc513819413)

[3.1 Diagram 11](#_Toc513819414)

[3.2 Description 12](#_Toc513819415)

[3.2.1 Get the created and mapped tags 12](#_Toc513819416)

[3.2.2 Read the tag value from I-Field server 12](#_Toc513819417)

[3.2.3 Read the tag value from ECC PI Server 12](#_Toc513819418)

[3.2.4 Compare Values and update tag flag based on the comparison result 12](#_Toc513819419)

[SECTION 4: Database Dependencies 13](#_Toc513819420)

[SECTION 5: Service App.Config 14](#_Toc513819421)

[SECTION 6: Installer 15](#_Toc513819422)

# Summary

This document describes the development manual of the PI Tag Value Checker part of the I-Field Tags Synchronization System Project at ECC, Saudi Aramco

This Manual considers a Visual Studio 2017 installed as a pre-requisite.

# Architecture

This section will visualize and describe how the code is structured for this service.

Each module represented is a standalone .NET project which are connected together via reference.

The main objective of this architecture is to provide a scalable and flexible solution implemented in multiple projects each with a specific role and concern.

## Diagram

**Windows Service**

**ECCPITagValueChecker\_Service**

**ECC\_DataLayer**

**.NET Class Library**

Oracle DB

**ECC\_Services\_Layer**

**.NET Class Library**

PI AF SDK

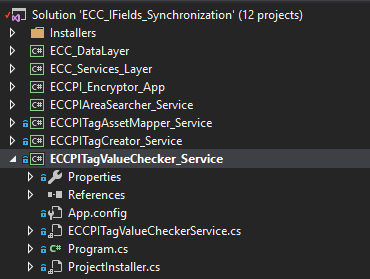
PI AF/

PI Data Archive

## ECCPITagValueChecker\_Service - Windows Service

This is a windows service which is the main output project that will be installed on the server.

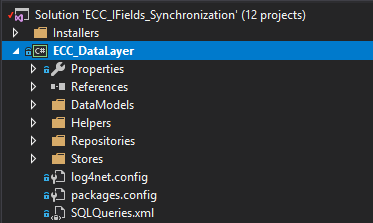
The main role of this module is to wrap all the logic implemented inside a windows service and handle the execution of the process.



## ECC\_DataLayer - Data Layer

This is a .NET class library responsible for communicating with database (mainly Oracle).

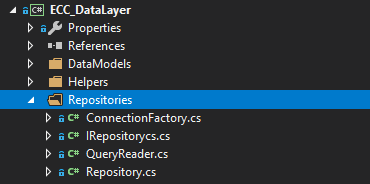
The data layer contains all the methods that are used by the service to Get/Update/Write data in database, also it contains the data models that represents the tables in the database.



Here is how the data layer is composed:

### Repositories

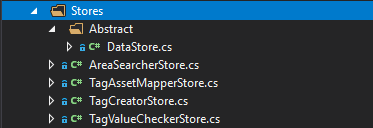
This is the repository which contain the classes and methods that Open/Close connection with the database, Execute a query (sync or async) using dapper ORM and Read a query from the ***SQLQueries.xml*** file.



### Stores

This namespace contains the implementation of the queries used, each class is for a specific windows service and contains all the methods that being used for executing the queries.

All stores inherit the ***DataStore.cs*** which implements some generic methods related to all inherited stores.



### Helpers

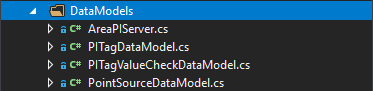
This namespace implements the general functionalities that could be used by any module or class in the project such like Encryption/Decryption of text or Logging.



### DataModels

It’s a representation of the tables used in the queries (specifically the SELECT queries) to manipulate from within the code.

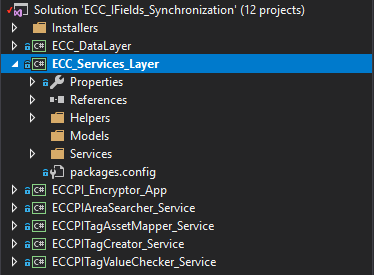
The queries results are mapped into these models using Dapper ORM (.NET SDK).



## ECC\_ServicesLayer – Service Layer

It contains all the business logic behind the service and communicate with PI Data Archive/ AF via SDK and also communicates with the ECC\_DataLayer when needed.

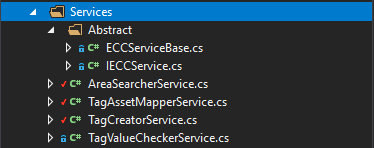
All the services required for the I-Field Tags Synchronization project are implemented inside this .NET class library.



Here is how the services layer is composed:

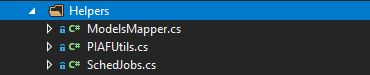
### Services

This namespace contains the implementation of the business logic of all services, each service is represented with a class where it contains all the workflow.



### Helpers

Contains the generic functionalities that are being used by all the services such like Objects Mapping, Communication with PI or AF via PI AF SDK and Service Job scheduling.



# Workflow

**ECCPITagValueChecker** is responsible of checking and validating the tags values from I-Field and ECC PI Server, this service should get the created and mapped tags and compare the values recorded at the origin I-Field server and ECC PI server.

## Diagram

Below a diagram representing the workflow.

Compare Values

Get the created and mapped tags

Schedule next execution date/time

Read the tag value from I-Field server

Read the tag value from ECC PI Server

Update tag flag based on the result

## Description

### Get the created and mapped tags

Query the created and mapped tags from the ECCPI\_AF\_WELL\_FOUND\_TAGS.

### Read the tag value from I-Field server

Search for the tag in the appropriate I-Field server by AREA\_TAG\_NAME and get the value 30mins before the service start time.

### Read the tag value from ECC PI Server

Search for the tag ECC PI serer by ECCPI\_EXST\_TAG\_NAME and get the value 30mins before the service start time.

### Compare Values and update tag flag based on the comparison result

Compare the values and set the PI\_TAG\_FLOW\_FLG to ‘Y’ if matching and ‘N’ if not.

# Database Dependencies

The table contains the main columns/flags that are being updated after the execution of the service.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table | Column Name | Type | | Usage | |
| ECCPI\_AF\_WELL\_FOUND\_TAGS | PI\_TAG\_FLOW\_FLG | Char | Used to identify if the tag values are matching between the ECC PI server and I-Field server. | |

# Service App.Config

This file is located under the **ECCPITagValueChecker\_Service** windows service and it contains the below keys.

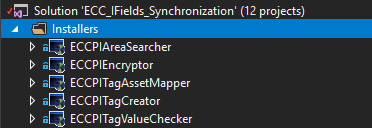
|  |  |
| --- | --- |
| Key Name | Description |
| ECC\_PI\_ServerName | ECC PI Data Archive server name or collective |
| ECC\_AF\_ServerName | ECC AF server name |
| ECC\_DB\_DataSource | Oracle data source name |
| ECC\_DB\_UserId | Oracle user Id |
| ECC\_DB\_Password | Encrypted oracle user password generated from Encryption tool |
| RUN\_FREQUENCY | The running frequency of the service.  Example: Daily, Weekly, Monthly or Yearly |
| RUN\_HOUR | Time of scheduled job in Hours |
| RUN\_MINUTE | Time of scheduled job in Minutes |

# Installer

This section describes where the installers are located inside the source code and how to generate an installation file for windows service.

The installer is a .NET Setup Project configured to build the windows service output and pack it into an installer file (.msi and .exe file).

Setup project is located under **Installers/** directory, there is located the setup project of all services each in a separate project as shown in the below image.



Once the developer has changed in the service code regardless in which project the changes has been made, a new installer should be provided to apply those changes.

In order to do that follow the below steps:

1. Locate the **ECCPITagValueChecker\_Service** Setup Project under **Installers/** directory
2. **Right click** and then click on “**Rebuild**”
3. Wait until the build is done then **Right click** on the project and click on “**Open folder in file explorer**”
4. Locate the installer under **Debug/** directory and double click on one of the installers created (.msi or .exe) to start the installation process.